FLORIDA SOUTHERN COLLEGE

Data Structures
CSC 3280

Fall 2021

Program 2: FSC Grade Book (Linked Lists)
Assigned: Tuesday, September 21, 2021
Due: Thursday, September 30th, 2021 by 11:59 PM

Purpose:
1. Practice the implementation and use of linked lists.

Read Carefully:

- This program is worth 7% of your final grade.

- **WARNING**: This is an individual assignment; you must solve it by yourself. Please review the definition of cheating in the syllabus, the FSC Honor Code, and the serious consequences of those found cheating.
  - **The FSC Honor Code pledge MUST be written as a comment at the top of your program**.

- When is the assignment due? The date is written very clearly above.
  - **Note:** once the clock becomes 11:59 PM, the submission will be closed! Therefore, in reality, you must submit by 11:58 and 59 seconds.

- LATE SUBMISSION: you are allowed to make a late submission according to the rules defined in the syllabus. Please see course syllabus for more information.

- **Canvas Submission**:
  - This assignment must be submitted online via Canvas.
  - Within IntelliJ, your project should be named as follows:
    - `FSCgradeBook`
    - And when you make the project, you will want to name the package as **`fscgradebook`** (all lowercase). Please do this exactly as typed.
  - You should submit a single **ZIP** file, which has **ONLY** your Java files inside it.
  - You should NOT submit the entire IntelliJ project.

Program 2: FSC Grade Book

Objective
Learn to implement the functionality of linked lists.

Program Description
For this program, you will implement a basic grade book for FSC. The grade book will include many courses, and each of these courses will be represented by a linked list of students who are registered for the courses. You will use File I/O to read commands from a file and then print the output to a file. Example commands are `ADDRECORD`, `SEARCHBYID`, `DISPLAYSTATS`, etc. Then, depending on the command, you will either add a student record, search for a student, display statistics, etc. But instead of printing to the console window (screen), you will print to an output file.
**Sample input and output files have been provided for you.*

For this program, you will create two classes:

- The first class is called `Student`. This class, `Student`, will be used to create objects of type `Student`. Each `Student` object will store all the needed information for one student in a specific course (ID, first name, last name, course number, exam grades, final grade, etc). These `Student` objects will be added (and deleted) from linked lists based on the commands from the input file.
- The second class is called `FSCcourseRoster`. This class will be used to create objects of type `FSCcourseRoster` (linked lists of students). This is the actual linked-list class. This is the class that will have the "head" and will include all the methods that are used to operate on a linked-list.

The first line of the input file is a positive integer, representing the number of courses to be maintained by the FSC Grade Book.

Each course will have its own linked list of `Student` objects. For example, if the integer, on the first line, is a 3, this means that the FSC Grade Book will maintain 3 courses, and this means that there will be 3 linked lists. If the integer on the first line is a 10, this means that the FSC grade book will maintain 10 courses, resulting in 10 linked lists. So the number of linked lists in the program is variable…meaning, it depends on the value read from the input.

Once you read the first `int` (`numCourses`), you will then make an array of `FSCcourseRoster` references, and you will use this input (`numCourses`) to make the correct size of the array.

As an example, if you want your array variable to be called `courses`, you can declare and create your array of `FSCcourseRoster` object references as follows:

```
FSCcourseRoster[] courses = new FSCcourseRoster[numCourses];
```

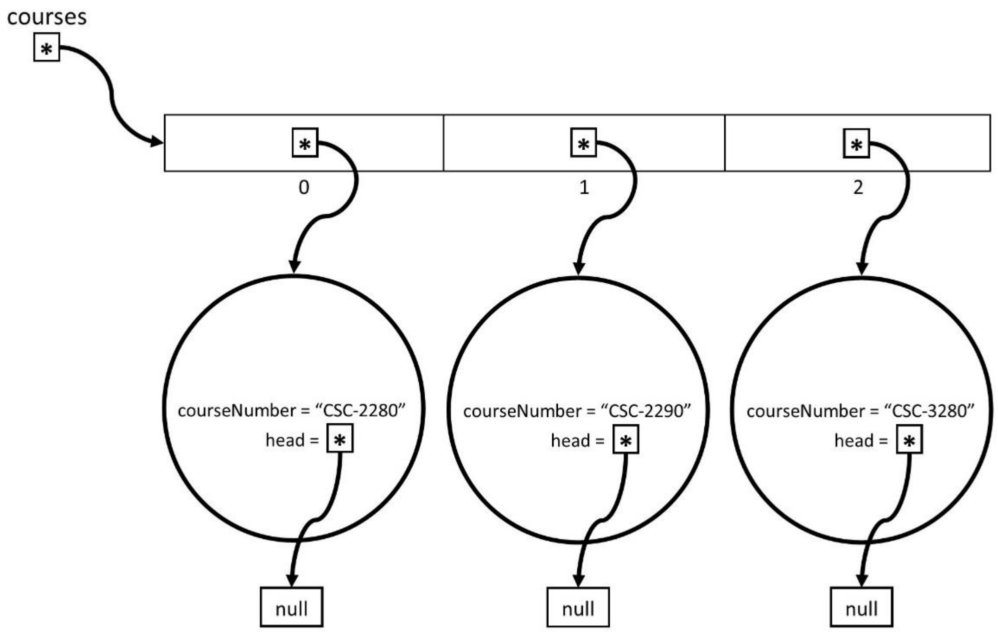
- **`numCourses`** is the value from line 1 of the input file

What does this line do? It creates an array of **references**, where each reference will ultimately point to an object of type `FSCcourseRoster`, which will be the linked list for that specific course. Note: until now, we have not created any objects of type `FSCcourseRoster`. You must now **loop** `numCourses` number of times and create a new `FSCcourseRoster` object for each index of the courses array. After you create an object of type `FSCcourseRoster`, you must then scan, from the input file, the course number and then save that value into the `FSCcourseRoster` object. Here is some code:

```
for (int i = 0; i < numCourses; i++) {
     courses[i] = new FSCcoursesRoster();
     courses[i].setCourseNumber(input.next());
}
```

So now, at this point, you have an array called courses, and the length of the array is `numCourses`, and each index of the array points to a linked-list object (`FSCcourseRoster`).

Here is a picture (for this example, we assume `numCourses` was 3 inside the file):


courses
*
*
0
*
1
*
2
courseNumber = "CSC-2280"
head = *
courseNumber = "CSC-2290"
head = *
courseNumber = "CSC-3280"
head = *
null
null
null

Now, from each of these `FSCcourseRoster` objects, you can add `Student` nodes to the list and modify the list.

Note: `Student` objects are only created when we see the command `ADDRECORD`. At that time, a new `Student` object will be created and the reference for that object will be saved in the appropriate linked list. (you simply call the `insert()` method of the linked list).

The Commands to be implemented are as follows

1. **ADDRECORD**

   This command will be followed by the following information in the input file: a String representing the course number (ex: CSC-3280), student ID (a non-negative integer), first name, last name, and then three exam grades, each being a non-negative integer. When you process this command, you should make a new student object, and then you should scan, from the file, the course number, the student ID, the first and last name of the student, and the three exam grades (see output). All of this information should be stored inside the newly created Student object. Next, the final grade and letter grade of the student will be calculated and saved into the Student object. The final grade is calculated as follows: *Exam 1 is worth 30%, Exam 2 is worth 30%, and the Final Exam is worth 40%.*

   Finally, you must insert this object into the appropriate course linked-list. How do you know which linked-list to save the Student node into? Each Student record will have a course number. So you must loop through the courses array and check the value of data member `courseNumber`. If `courseNumber` matches the course number read from the input file, then this is the correct linked-list. Now, you must invoke the insert method to insert the `Student` object into the list. It is possible that the student is being added to a course that does not exist. Meaning, the course number is not found in any of the `FSCcourseRoster` objects. In this case, print an error message (see output).

   Example:

   If the following line was in the input file:
   ```
   ADDRECORD CSC-3280 111 Alan Turing 90 85 94
   ```

   This would be the output:
   ```
   Command: ADDRECORD
   Alan Turing (ID# 111) has been added to CSC-3280.
   Final Grade: 90.10 (A).
   ```

**So the final grade of 91.10 was calculated by: 90*(0.3) + 85*(0.3) + 94*(0.4)

2. **DELETERECORD**
   This command will be followed by a student ID, a non-negative integer. If the record is found in any linked-list, it should be deleted and a message printed to output file (see sample output file). You must delete the student from ALL courses they are registered in. If the ID is not found in any linked list, an error message should be printed instead (see sample output file).

3. **SEARCHBYID**
   This command will be followed by a student ID, a non-negative integer. If the record is found, it is printed to the file (see sample output file). If it is not found, an error message should be printed instead (see sample output file).

4. **SEARCHBYNAME**
   This command will be followed by a student first and last name. If the record is found, it is printed to the file (see sample output file). If it is not found, an error message should be printed instead (see sample output file).

5. **DISPLAYSTATS**
   This command will be followed by a String, which will be a course number or "ALL". If a course number is read, you should print the stats for that specific course. If "ALL" is read, you should print the stats for all courses (see output). You are guaranteed that the course number will be valid.

6. **DISPLAYSTUDENTS**
   This command will be followed by a String, which will be a course number or "ALL". If a course number is read, you should print the student list for that specific course. If "ALL" is read, you should print the student list for all courses (see output). You are guaranteed that the course number will be valid.

7. **QUIT**
   This command will not have any other information after it (in the input file). When you scan this command, the program should print a message to the file and then quit.

<u>Input File Guarantee/Promise</u>
You are <u>guaranteed</u> that no other commands will be in the input file. You are also <u>guaranteed</u> that the input will be correct, which means you do not have to worry about mistakes in the input. Finally, you are <u>guaranteed</u> that each ADDRECORD command will

be independent of all other commands (there will not be duplicate ADDRECORD commands in the input).

Input File & Output File

You must read from an input file called **FSCgradeBook.in**. The first line will have a positive integer, *k*, representing the number of courses (linked-lists) to create and maintain. The next *k* lines will have one String per line, representing the course number. The remaining lines will each start with a command. Each command will have the required information, on the same line, as described above in the write-up.

Sample Input & Output File

You must print your output to an output file called **FSCgradeBook.out**. We have provided you a sample input with matching output (see the 7z file you downloaded).

*****WARNING*****

Your program MUST adhere to the EXACT format shown in the sample output file (spacing capitalization, use of dollar signs, periods, punctuation, etc). The output files are very large, dictating the use of text comparison programs to compare your output to the correct output. If, for example, you have two spaces between words in the output, when there should be only one space, this will show up as an error even though you may have the program logic correct. This will have to manually corrected by me in order to effectively grade your program, and these corrections will result in deductions, which is why this is being explained in detail. Minimum deduction will be 10% of the grade, as I will be forced to go to text editing of your program in order to give you an accurate grade. Again, your output must adhere exactly to the formatting depicted in the sample output. Use of a text comparison program makes this very easy for you. While I recommend the downloadable version of WinMerge for PC users, there are numerous online text comparison applications, including Mergely, which many of you used previously.

Grading Details

Your program will be graded upon the following criteria:

1) Adhering to the implementation specifications listed on this write-up.
2) Your algorithmic design.
3) Correctness.
4) **Use of Linked-Lists. If your program does not use linked-lists, you will get a zero.**

5) The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (We're not kidding here. If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)
6) Compatibility to the **newest version** of **IntelliJ IDEA**. (If your program does not compile in IntelliJ, you will get a large deduction from your grade.)
7) Your program should include a header comment with the following information: your name, **email**, course number, section number, assignment title, and date.
8) Your output MUST adhere to the EXACT output format shown in the sample output file.

Deliverables

You should submit a zip file with THREE files inside (**only** the Java files):

1. `Student.java` (this is the class to make individual nodes)
2. `FSCcourseRoster.java` (this is your Linked List)
3. `FSCgradeBook.java` (this is your main program)

***These three files should all be INSIDE the same package called **fscgradebook**. If they are not in this specific package, you will lose points.

NOTE: your name, ID, section number AND EMAIL should be included as comments in all files!

*****Hint*****

Hint 1: Getting and setting the exam grades of a student may be tricky at first. So here is a bit of code from the `addRecord()` method, from which we scan and save the student information:

```
for (int i = 0; i < 3; i++) {
    tempStudent.setExamGrades(input.nextInt(), i);
}
```

So as you can see, we are saving the three exam grades into the `examGrades` array. How are we doing this? We invoke the `setExamGrades()` method with two parameters:

1. The first parameter is the grade received from the input file
2. The second parameter is a counter variable, `i`. So when '`i`' is zero, this means that the inputted exam grade will get "set" at index zero of the `examGrades` array. When '`i`' is 1, the inputted exam grade will get "set" at index 1…and so on.

Hint 2: calling methods of the linked-list object (`FSCcourseRoster` object):

For example, when you read the command, `ADDRECORD`, you must make a new Student object and then insert that student into the correct linked-list. How do you do this? You must loop over the number of linked-list objects and compare the `courseNumber` of the new student with the `courseNumber` of the `FSCcourseRoster` object. Here is some code:

```
for (int i = 0; i < courses.length; i++) {
      if (tempStudent.getCourseNumber.equals(courses[i].getCourseNumber()){
            courses[i].insert(tempStudent);
      }
}
```

This code will insert the `tempStudent` object into the correct linked list. And BEST of all is that you needn't worry about "how" to do the `insert()` method, as that is already done for you; it's merely the included `insert()` method of linked lists!

Suggestions

1. Start early.
2. Make the skeleton of your main program by using the `IOexample.java` file as a template.
3. Modify the commands to match the commands for this programming assignment.
4. Make methods for each command, and send to the methods the input and output reference values (`Scanner` and `PrintWriter`).
5. Start by merely READING the input file and then printing that EXACT SAME input file to the output file, thus verifying that you have read everything perfectly!
6. Make the `Student` class.
7. Make the `FSCcourseRoster` class.
8. For the `FSCcourseRoster` class, use the LinkedList.java code (from the website) as a reference.
9. Add methods to the `FSCcourseRoster` class and to your main `FSCgradeBook` class.

And, as always, the final suggestion: Start early.

<u>UML Diagrams for Three Classes</u>

Student
Data Members private String courseNumber private int ID private String firstName private String lastName private int[] examGrades private double finalGrade private char letterGrade private ***static*** int numStudents private Student next
Operations/Methods FSCmember() // one or more Constructors ALL getter and setter methods

FSCcourseRoster
Data Members private Student head private String courseNumber
Operations/Methods public boolean isEmpty() public void setCourseNumber(String courseNumber) public String getCourseNumber() public boolean searchID(*parameters here*) private boolean searchID(*parameters here*) public boolean searchName(*parameters here*) private boolean searchName(*parameters here*) public Student findNode(*parameters here*) private Student findNode(*parameters here*) public void insert(*parameters here*) private Student insert(*parameters here*) public void delete(*parameters here*) private Student delete(*parameters here*) public void printRoster(*parameters here*) private void printRoster(*parameters here*) public void printStats(*parameters here*) private void printStats(*parameters here*)

- The methods shown above are just suggestions.
- You can use these methods.
- You can make your own.
- The choice is for you.

<u>NOTE</u>:
You will use the `Student` class to create nodes of type `Student` (these are the nodes of the several linked-lists), and you will use the `FSCcourseRoster` class to create nodes of type `FSCcourseRoster` (these are the objects of the actual linked-lists).

*All members of the Student class must be private. As a result, you must use accessor and mutator methods (get and set methods) to access and change/set the values.

**Pay close attention to the `getExamGrade`() and the `setExamGrade`() methods. For the `getExamGrade`() method, you send one parameter, the index of the exam you want to get. For the `setExamGrade`() method, you send two parameters: the grade of the exam and the index where you want to set it. Of course, this was *my* implementation; you may certainly choose to implement your solution differently!